SOFIA's Choice: Scheduling Observations for an Airborne Observatory

Jeremy Frank, Elif Kurklu*
Computational Sciences Division
NASA Ames Research Center, MS 269-2
{frank,ekurklu}@email.arc.nasa.gov
Moffett Field, CA 94035

Abstract

We describe the problem of scheduling observations for an airborne astronomical observatory. The problem incorporates complex constraints relating the feasibility of an astronomical observation to the position and time of a mobile observatory, as well as traditional temporal constraints and optimization criteria. We describe the problem, its proposed solution and the empirical validation of that solution.

Introduction

The Stratospheric Observatory for Infrared Astronomy (SOFIA) is NASA's next generation airborne astronomical observatory. The facility consists of a 747-SP modified to accommodate a 2.7 meter telescope. SOFIA is expected to fly an average of 140 science flights/year over it's 20 year life time. The SOFIA telescope is mounted aft of the wings on the port side of the aircraft and is articulated through a range of 20 to 60 degrees of elevation. The telescope has no lateral flexibility; thus, the aircraft must turn constantly to maintain the telescope's focus on an object during observations. A significant problem in future SOFIA operations is that of scheduling Facility Instrument (FI) flights in support of the SOFIA General Investigator (GI) program. GIs are expected to propose small numbers of observations, and many observations must be grouped together to make up single flights. Approximately 70 GI flight per year are expected, with 5-15 observations per flight. The scope of the flight planning problem for supporting GI observations with the anticipated flight rate for SOFIA makes the manual approach for flight planning daunting. There has been considerable success in automating the scheduling of astronomical observations in a variety of contexts, ranging from ground-based telescopes (Bresina 1996) to the Hubble Space Telescope (Johnston & Miller 1994). In this paper, we describe the application of automated scheduling techniques to this problem.

A Quick Astronomy Lesson

The principal constraints in the SFPP are those governing the visibility of astronomical objects and the motion of the aircraft. As we shall see, these constraints interact in complex ways, leading to an interesting scheduling problem. We will follow the conventions in Meeus (Meeus 1991) in determining the visibility conditions for an astronomical object and the equations of motion of the aircraft that follow. Meeus uses the following definitions:

- α is the Right Ascension (RA) of the object to be observed (similar to Longitude on Earth).
- δ is the Declination (Dec) of the object to be observed (similar to Latitude on Earth).
- ϕ is the Earth latitude at which the observation occurs. Positive latitudes are in the Northern hemisphere.
- L is the Earth longitude at which the observation occurs. Positive longitudes are measured West from Greenwich.
- θ is the "time" [1] at which the observation is performed.
- h is the elevation angle of the object relative to the horizon at the location of the observation. Positive elevations are above the horizon.
- A is the azimuth of the observation at the location of the observation. Azimuth is measured in degrees West, assuming that 0 degrees is due South.

Suppose all quantities except h and A are fixed. We can solve for h and A as follows:

$$H \equiv \theta - L - \alpha \tag{1}$$

$$\sin h = \sin\phi \sin\delta + \cos\phi \cos\delta \cos H \tag{2}$$

$$\tan A = \frac{\sin H}{\cos H \sin\phi - \tan\delta \cos\theta} \tag{3}$$

The telescope has minimum and maximum elevation limits, and so there may be times when an object is not visible, even if it is above the horizon. Figure 1 shows the visibility

*QSS Group Inc.

[1] Time, in this case, refers to Greenwich Sidereal Time, which is related to how long it takes the Earth to orbit the Sun, as opposed to local time, which is the amount of time it takes the Sun to reach the same position in the local sky.

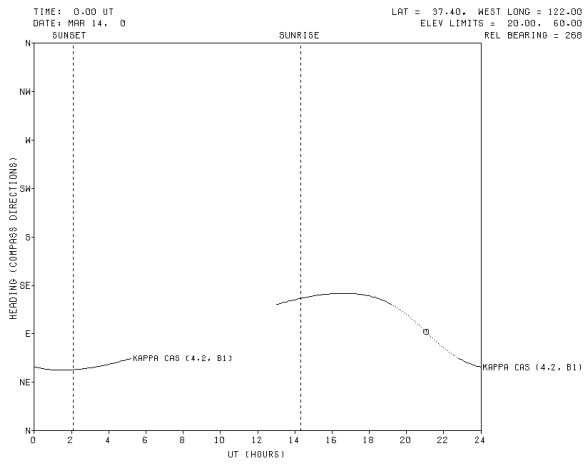

Figure 1: Visibility and azimuth of an object as seen from Moffett Field, CA. as a function of time. The gap in the graph indicates that the object elevation is below 20 degrees, and the dotted line indicates that the object elevation is above 60 degrees. Notice how the azimuth can vary greatly over periods of time as short as half an hour.

windows for an object over a 24 hour period when viewed from a fixed position.

Since the telescope has no lateral flexibility, the aircraft must turn continuously during an observation in order to keep the object in view. Figure 1 also shows the direction that the aircraft must fly in order to observe the object over time, at a fixed position. The aircraft is continuously in motion during observing. Consequently, the position of the aircraft after completing an observation is a complex function of the object's coordinates and the observation's position and the start time. Beginning with the previous equations describing the azimuth of the observation, the equations of motion of the aircraft can be derived. Suppose the *ground speed* of the aircraft is V, the *relative bearing* [2] of the telescope is r, and that the Earth is a sphere whose radius is a. Assume that the aircraft moves at velocity V in such a way that the angle between the direction of motion and the object's azimuth is constantly r. Then the equations of motion are

$$\frac{d\phi}{d\theta} = -\frac{V\cos(A-r)}{a} \tag{4}$$

[2]The direction the telescope points relative to the direction of motion of the aircraft.

$$\frac{dL}{d\theta} = -\frac{V\sin(A-r)}{a\cos\phi} \tag{5}$$

where we solve for A using Equation 3.

Flight Plans

We use a simple model of the aircraft state when planning flights. The aircraft can be on the ground, taking off, landing, performing a turn, performing a flight leg, or performing a dead leg. A *flight leg* is a period of time during which an observation is taking place. We assume that the target remains fixed throughout the flight leg. The aircraft's final position after a flight leg is determined by solving Equations 4 and 5. A *dead leg* is a leg during which no observation is taking place. Dead legs are needed to reposition the aircraft in order to enable observations; for instance, if an object is out of elevation range, flying a short dead leg can move the aircraft to a position where the object elevation is within the range allowed by the telescope. The takeoff and landing legs are also assumed to be dead legs. The aircraft's final position is found by assuming that the ground track is a Great Circle on the surface of the sphere whose distance and heading are determined by the planner. Takeoff and landing activities are special dead legs indicating the beginning and ending of the flight plan. Takeoff terminates at one of a set of waypoints associated with the airport, and similarly landing must begin at one of these waypoints. We assume that each leg is separated by a turn, and that the aircraft turns at a rate of 180 degrees every 2 minutes (the Standard Rate Turn for a 747).

Problem Statement

We assume that a number of individual observation requests from GIs have been accepted for observing and assigned prioritized. The input to the SFPP is defined as follows:

- A set of observation requests, each consisting of the Right Ascension (RA) and Declination (Dec), observation duration, priority, earliest start time and latest end time.
- A flight date.
- A maximum flight duration.
- A flight horizon (i.e. earliest takeoff time and latest landing time).
- A designated takeoff and landing airport (which need not be the same).

The objective is to find a flight plan that maximizes the summed priority of the observations of the observations performed. Since it is intractable to find the best possible plan, we will relax this constraint and limit ourselves to searching for *good* plans that perform many observations of high priority.

The scheduling problem requires making the following choices:

- Which observations to perform.
- The order in which the observations are performed.
- The takeoff time and waypoint.
- Whether or not dead legs are performed.

- Dead leg duration and heading.
- Landing waypoint.

ForwardPlan(K, O, H, S, P, F**)**
$Q = \emptyset; P' = \emptyset$
repeat S times
  choose a value $h \in H$
  $P' = P'||h$
  $O' = O$
  P'=Lookahead(K, O', P', F)
  $Q = Q \cup$ (h,FullEvaluate(P'))
h=Select(Q)
$P = P||h$
while O not empty
  $Q = \emptyset; P' = \emptyset$
  for each unscheduled observation $o \in O$
    if Feasible(o, P, F)
      $P' = P||o$ # o includes dead leg, if necessary
      P'=Lookahead(K, O', P', F)
      $Q = Q \cup$ (o,FullEvaluate(P'))
  if Q not empty
    o' =Select(Q)
    $P = P||o'$
    remove o' from O
  else
    $O = \emptyset$
    break
return P
end

Figure 2: A sketch of the SFPP Flight Planning Algorithm.

Algorithm Description

A quick look at the search space reveals the scope of the problem. Notice that there are no constraints on the choice of dead leg heading or duration. If time were modeled as a continuous quantity, then the search space would be mathematically infinite; even if time and heading were modeled as integers, the search space would be quite large. Without considering these problems, the search space is "merely" combinatorial.

We observe that it is necessary to know both *where* an observation begins and *when* an observation begins in order to know where the aircraft is after the observation ends. This can be seen due to the fact that Equations 4 and 5 take the azimuth A as a parameter, which in turn is a function of the position of the aircraft and the time the observation is performed. It is also necessary to know where and when the observation begins to determine whether or not the observation is feasible, since this requires ensuring that the elevation constraints are not violated during the observation. Since we need to know the location of the aircraft and the time of the observation in order to evaluate candidate flight legs, we use a progression planning algorithm that simulates the flight from takeoff to landing. The algorithm proceeds

Lookahead(K, O, F, P**)**
repeat K times
  $Q = \emptyset$
  while O is not empty
    for each unscheduled observation $o \in O$
      if Feasible(o, P, F)
        $Q = Q \cup$ (o,LookEvaluate($P||o$))
    if Q not empty
      o' =Select(Q)
      $P = P||o'$
      remove o' from O
    else break
return P
end

Figure 3: A sketch of the Lookahead phase of the SFPP Flight Planning Algorithm.

by evaluating the utility of performing each of the unscheduled observations next in the plan using a combination of lookahead and heuristics. Once the observations are evaluated, one is chosen to perform next, and the process repeats until there are no feasible observations left. The resulting algorithm, called ForwardPlan, is shown in Figure 2. In the algorithm descriptions, K is the lookahead distance, O is the set of observations, S is the number of start time samples, P is the schedule, H is the set of possible takeoff times, F is the flight duration.

The algorithm can potentially perform a large number of flight leg construction steps, as well as searching for dead legs to enable observations. Recall that constructing a flight leg requires solving Equations 4 and 5. To minimize these costly operations, instead of using a full lookahead, we use a second heuristic (called LookEvaluate() to distinguish it from the FullEvaluate() function used in ForwardPlan) to guide the selection of observations used in the lookahead phase. The heuristic evaluations are used as the input to a stochastic selection algorithm.

Figure 4 shows the search process. ForwardPlan evaluates each observation as a candidate for extending the plan by first assuming the plan is extended by an observation, then performing lookahead. The lookahead phase builds the "best" extension of this plan using the LookEvaluate() heuristic. Once this has been done for each observation, these plans are evaluated using the FullEvaluate() heuristic. The results of this evaluation are used to rank each observation, and these ranks are used to choose the next observation in the plan.

We now analyze the computational complexity of ForwardPlan. We will analyze the complexity in terms of calls to Feasible(), which comprises a number of flight leg and dead leg construction steps. Let us assume that the start time has been selected. Let N be the number of observation requests, let K be the lookahead depth, and let M be the

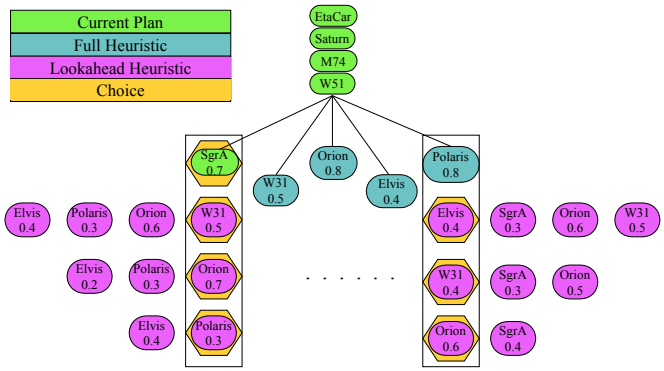

Figure 4: ForwardPlan's search. At each step, all feasible observations are considered as the next step in the plan. An extension of the plan is built using the LookEvaluate heuristic, and the FullEvaluate heuristic is used to determine which observation to perform next.

maximum number of observations that can be in any flight plan. Then the algorithm makes $O(N^2KM)$ calls to Feasible(). The proof is as follows: For each call to Lookahead(), we construct a plan with at most K steps, in which we call Feasible() for each of the observations not scheduled during the lookahead phase. This number is bounded above by N. Thus, each call to Lookahead() costs $O(NK)$ calls to Feasible(). A plan is assumed to have at most M steps. For each of these steps, we call Lookahead at most N times, since we must consider extending the plan by each unscheduled observation. In addition, for each of these steps, we make at most N calls to Feasible() to decide which observations must be evaluated. Thus, we have $O(N^2KM)$ calls to Feasible(). The cost of evaluating S start times is S calls to the Lookahead() function, which is dominated by the rest of the algorithm. Note that the maximum number of observations that can be in a flight plan, M, is unknown, and could be $O(N)$ in some cases; thus, in the worst case, the algorithm is $O(N^3)$.

Constructing Dead Legs

The feasibility check Feasible(o, P, F) attempts to extend P by adding observation o to the end of P. This requires determining whether or not the visibility constraints are violated during the flight leg, and ensuring that the aircraft can fly to the landing airport after the flight leg is finished. If the visibility constraints are violated, it may be possible to construct a *dead leg* to reposition the aircraft or delay the observation. For example, if the object is just below the horizon and is rising, either flying towards the object or delaying the observation by a short time can make the observation feasible. It is

Feasible(o, P, F**)**
\# D is the maximum dead leg duration
\# I is the dead leg increment, E is the heading increment
Construct flight leg for o
Construct dead leg to landing airport
if the flight duration $< F$ and object is visible
  return true
d'= duration of dead leg enabling o by changing latitude
e'= heading of dead leg enabling o by changing latitude
if $d' > D$
  return false
for $d = d'$ to 0 by $-I$
  noDeadLeg=**true**
  \# Search headings +/- 90 degrees from e'
  for $e = e' - 90$ to $e' + 90$ by E
    Fly dead leg specified by d, e
    Construct flight leg for o
    Construct dead leg to landing airport
    if flight duration $< F$ and object is visible
      noDeadLeg=**false**
      deadLeg= (d, e)
      break
  if noDeadLeg==**true**
    break
$o =$ deadLeg $||o$
return true

Figure 5: A sketch of the Feasible check of the SFPP Flight Planning Algorithm.

not always possible to construct a dead leg to observe an object. For example, if an object is setting and the aircraft is at low or middle latitudes, then the speed of the Earth's rotation exceeds the aircraft's maximum speed, so the aircraft can't catch up to the object. The search space for this consists of all possible headings and dead leg durations, which is very large. We restrict this search by limiting the maximum dead leg duration D, constraining the dead legs to be multiples of a constant value I, and restricting the possible heading changes to be multiples of a constant value E. These are tunable parameters of ForwardPlan. The Feasible() check first searches for a dead leg that restricts the headings to either due North or due South. A shorter dead leg enabling the observation may exist; for instance, if the object is below the minimum elevation and rising, flying towards it will minimize the time until it is visible. So we attempt to find a dead leg of shorter duration. We do not need to search dead legs that change the latitude in the reverse direction, as they are guaranteed to be longer than the dead leg we just discovered. As soon as we find a dead leg duration which could not enable the observation, the procedure halts.

As can be seen from the sketch of this procedure, each call to Feasible() results in at most $\frac{2D}{I} + \frac{180D}{EI}$ constructions of flight legs, enabling dead legs, and dead legs to return the aircraft to the landing airport. However, in many cases the actual number may be much smaller.

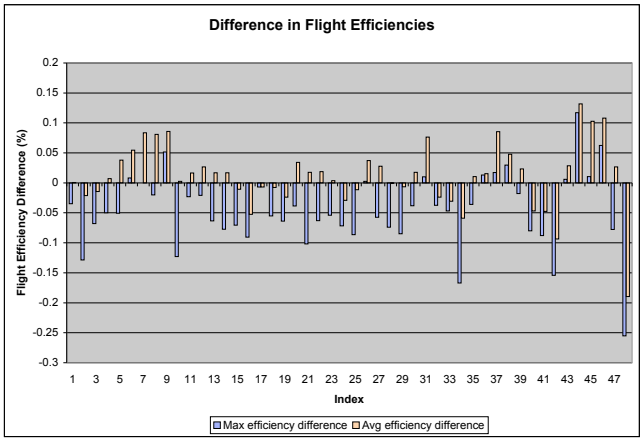

Figure 6: Difference in flight efficiencies between archived schedule and schedules we find. This plot shows the difference between the archived flight efficiency and both the maximum efficiency and the average of all efficiencies maximizing the number of observations scheduled.

Heuristics

Heuristics play two roles in the ForwardPlanner algorithm. They are used to decide which observations are added during the lookahead phase (the call to LookEvaluate()), and they are used to commit to the next observation in the plan based on the results of the results of lookahead (the call to FullEvaluate()). We view both heuristics as a mapping from a flight plan P to a real number. In this section, we describe the heuristics in detail.

A good flight plan is one that observes many high valued observation requests, but takes as little time as possible. Indirectly, this means minimizing the amount of dead leg time. A heuristic should also pay attention to how much time is needed to get the aircraft to the landing airport; if observations naturally carry the aircraft towards the landing airport, that may reduce dead leg time to land after observations are completed. However, plans that "loiter" over the landing airport may not enable the most important observations.

We have identified four *features* of a flight plan that serve as the input to heuristics:

1. Importance: the summed priority of the observations performed in the flight plan
2. Efficiency: the summed duration of flight legs divided by the total flight duration
3. Dead leg home distance: the amount of time required to fly to the landing airport
4. Turn amount: the total degrees of heading changes between flight legs

Some of these features work in opposition to each other. Relying on importance more than efficiency may lead to poor choices when a high priority observation is very inefficient. Relying on efficiency more than importance may lead to poor choices when high priority observations do not lead to great inefficiency. For this reason, we associate with each of these features a real valued weight between 0 and 1. The heuristics then map a flight plan into a real number by summing the weighted value of each feature. We can thus express a number of heuristics that play these features off each other in an attempt to identify a good heuristic. ForwardPlan is implemented so that LookEvaluate() and FullEvaluate() can use different weigts and implement different heuristic functions.

Analysis

In this section, we analyze the performance of the flight planner algorithm. We first describe the source of our problem instances. We then discuss some preliminary experiments to determine good parameter settings for the algorithm parameters. We then present experiments to determine the effectiveness of the algorithm using the parameter values we felt were most suitable on two different classes of flight planning problems.

Sample Problems

The SOFIA observatory is the successor to the Kuiper Airborne Observatory (KAO), which performed infrared astronomical observations between 1974 and 1995. It is expected that SOFIA will be used to observe many of the same objects that were observed with KAO, and the telescope elevation of SOFIA is similar to that of KAO. NASA Ames Research Center has many years of archived flight plans that were executed aboard KAO. Since SOFIA is not yet operational, we obtained from KAO astronomers a representative set of flights that we used to construct test instances. These flights were flown from Moffett Field, CA; Honolulu, HI and Christchurch, NZ between 1988 and 1995, and occurred between the months of April and November. The archived flight plan data contains only a specification of the flight plan that was actually executed. It contains no indication of what set of requests were considered when making the plan. Furthermore, the flight plans reflect considerations such as restricted airspace and predicted winds that our planner cannot yet handle. Nevertheless, they provide a reasonable benchmark to compare the performance of our planner.

We created two sets of problem instances using the archived flight plans. The first set of problem instances were designed to test the basic performance of the algorithm. These problem instances were created using a single archived flight as the basis of each instance. As such, we refer to them as the Single Day Instances. Each observation was converted into a request to observe the object for the same amount of time it was observed in the archived plan. We gave all observation requests the same priority [3]. The

[3]The KAO astronomers had indicated informal object priorities

flight was requested to take off and land at the same airports used in the archived plan. The flight was requested to take less than 1.1 times as long as the archived flight. Finally, the flight was requested to take off no earlier than 30 minutes before the archived takeoff time, and land no late than 30 minutes after the archived landing time.

Figure 7 lists some salient characteristics of the Single Day Instances. We tabulate the number of observations, the archived flight duration, and the airport. In only one case did the takeoff and landing airports differ; in this case, the flight plan began at Moffett Field and ended at Honolulu. The instance numbers in this table will be used to present results of our algorithm later in the paper.

The second set of problem instances was designed to push the planner by ensuring that not all observations could be scheduled. These instances were constructed from the first set by merging requests for different flights. Thus, we refer to them as the Multiple Day Instances. We merged flights that were originally executed within one or two months of each other during the same calendar year. With one exception, we only merged flights from the same airports. We adjusted the flight duration to be the maximum of the flight durations of the Single Day Instances used to construct the new instance. In addition, we adjusted the earliest takeoff time to be the minimum of the earliest takeoff times of the Single Day Instances, and the latest landing time to be the maximum of the latest landing times. We then generated one instance with the set of all of the requests and the new duration, takeoff and landing times, for each date corresponding to the contributing Single Day Instances. Due to the variability in the number of observations in the Single Day Instances and number of flights taking place within a short period of time, the number of observations in these larger problems varies widely. Figure 8 describes the instances formed by combining problem Single Day Instances flown within one month of each other. Figure 9 describes the instances formed by combining problem Single Day Instances flown within two months of each other.

It is worth noting some features of our test problems. While the problem description admits varying observation priorities, our test problem instances all had identical priorities for observations. While the problem description allows for constraints on the legal times to perform an observation, our problems have no additional constraints beyond those imposed by the takeoff and landing time. Since the instances do not specify either winds or restricted flight zones, it is possible that shorter plans than the archived ones could be found by our planner. Finally, all of the problems have known solutions, and thus might be considered ”easy” for this reason. Our problem formulation strategy ensures that these solutions are disguised by increasing the search space in a manner consistent with our expectations on the actual flight planning problems.

for only a subset of the flight plans delivered to us, and had no basis to compare two flight plans with different objects except for total observing time.

Parameter Settings

We have implemented ForwardPlanner using EUROPA (Frank & Jónsson 2003), a constraint-based planning infrastructure that supports a wide variety of planning algorithms. We implemented the constraints to solve Equations 4 and 5 via Euler’s method (Ferziger 1981) and assume that the ground track is composed of a sequence of Great Circles approximating the true ground track.

We conducted preliminary experiments on a small set of problems from the Single Day Instances in order to choose good parameter values for the algorithm. We chose one flight each from Moffett Field, CA, Christchurch, NZ and Honolulu, HI to perform our initial experiments.

We considered the following parameters:

- Whether to include or omit each feature of the heuristic in the call to LookEvaluate().
- Whether to include or omit each feature of the heuristic in the call to FullEvaluate().
- Lookahead depth.
- Number of start times.
- The proportion of the time the heuristic is used to drive greedy selection as opposed to using the output of the heuristic probabilistically. We allowed different proportions for the Select() function calls in Lookahead() and in ForwardPlan().

Initially, we did not vary the parameters governing the search for dead legs. For these experiments, we restricted the maximum dead leg duration to 4 hours, the dead leg increment to 1 minute, and the heading increment to 7.5 degrees. We describe experiments varying the dead leg search parameters in a later section.

We searched for combinations of parameter values that ensured good performance on the subset of problems we used for these experiments. Good performance in this case meant that the parameter settings enabled the algorithm to find schedules in which all of the observations were performed, and the overall flight duration came as close as possible to the duration of the flight executed in the KAO archives.

Due to the large space of variations in the algorithm settings, we investigated only a small fraction of the possible combinations. Our results led to the following informal conclusions, in rough order of importance:

1. Employing a suitable lookahead distance was important to achieving good performance. A lookahead distance of 4 observations was achieved a good balance between computational cost and finding good flight plans. Note that this is roughly half the length of the average Single Day flight.
2. Using the heuristics with a small amount of stochasticity in the lookahead phase was beneficial. The heuristic was always used greedily in the observation selection phase. In retrospect, this may be because the archived flight plans had no dead legs apart from the takeoff and landing legs; thus, greedily choosing observations that the heuristics ranked highly might be the best course.

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Airport	H	H	H	H	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	MH
# Obs	9	9	10	10	7	8	8	6	10	8	8	6	11	10	8	9	10	8	8	8	9	9	6	8
Dur	437	432	440	441	460	495	500	515	610	470	600	390	432	437	445	439	440	449	448	441	440	442	293	435
Index	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Airport	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	N	N	N	N	N
Obs	7	4	7	6	7	9	8	11	10	8	7	7	7	3	9	8	8	8	4	10	8	8	8	8
Dur	440	316	443	440	443	451	442	443	437	448	438	380	385	232	447	442	441	441	192	495	470	460	465	460

Figure 7: Characteristics of Single Day Instances.

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Airport	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
# Obs	23	23	23	32	32	32	32	27	27	27	61	61	61	61	61	61	61	15	15	15	15	15	44	44
Index	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42						
Airport	M	M	M	M	M	M	M	M	M	M	M	N	N	N	H	H	H	H						
# Obs	44	44	44	58	58	58	58	58	58	58	58	24	24	24	38	38	38	38						

Figure 8: Characteristics of one-month span Multiple Day instances.

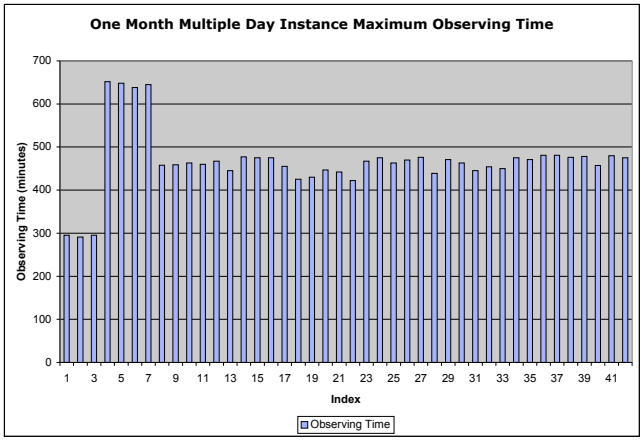

Figure 10: Maximum amount of observing time scheduled for the One Month Multiple Day Instances.

3. Examining 5 start times was sufficient to achieve good performance.

4. When evaluating observations in the lookahead phase, evenly weighting all 4 features of the heuristic achieved good performance. We did not analyze uneven weighting of the features.

5. When evaluating observations after lookahead, only Importance and Efficiency contributed to the heuristic. When weighted evenly, the algorithm achieved good performance; we did not consider uneven weightings of the features.

These settings were used in the experiments that are described in the following sections.

Single Day Instances

After settling on the parameters of the algorithm, we ran the flight planning algorithm with these parameter settings on all of the Single Day Instances. The goal of this experiment was to ensure that the parameter settings we found were not biased by the small number of problem instances. ForwardPlanner was able to schedule all of the observations for all but 3 of the instances; problems 31,37 and 46. When the algorithm was not able to perform all of the observations, it omitted only one observation.

Recall that the Single Day Instances limit the duration of the flight to be 1.1 times as long as the archived flight plan. We compared the flight efficiency of flights found by the planner. Figure 6 shows differences in the flight efficiencies of the archived plan and those found by our algorithm. In the figure we have plotted the difference between the efficiency of the archived flight and the best flight efficiency we found, and the difference between the archived efficiency and the average flight efficiency of those flights for which we were able to observe all of the objects. If the difference is positive, then the archived flight was more efficient than the flight (or the average) found by our algorithm. We see from the results that in most of the cases, the most efficient flight found by our algorithm was better than the archived flight efficiency. About half the time, the average efficiency of the flights found by our planner is better than the archived flight efficiency. Note that these results should be taken with a grain of salt, because the archived flights account for restricted airspace, which tends to lengthen flights, and nominal winds, which can either lengthen or shorten a flight.

The flights from the KAO contained no indication of relative object priorities, so the priorities of all observations were equally weighted. We performed a series of 10 experiments in which the priorities of the observations were randomly selected from the range 1 to 10. We then performed exactly the same experiment. In all cases, the performance was as good or better as in the case where all observations priorities were equally weighted.

Multiple Day Instances

We then ran the algorithm with the same parameter settings on the Multiple Day Instances. Since no archived flight

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Airport	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
# Obs	76	76	76	76	76	76	76	76	76	55	55	55	55	55	55	55	102	102	102	102	102	102	102	102
Index	25	26	27	28	29	30	31	32	33	34	35	36	37	38										
Airport	M	M	M	M	M	M	M	M	M	M	M	M	M	M										
# Obs	102	102	102	102	102	62	62	62	62	62	62	62	62	62										

Figure 9: Characteristics of two-month span Multiple Day Instances.

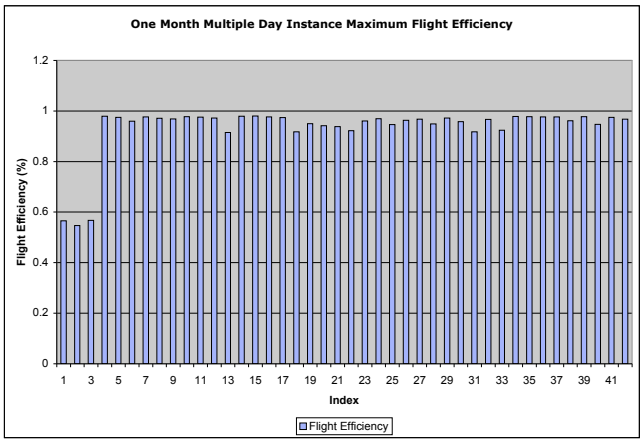

Figure 11: Flight Efficiencies for the One Month Multiple Day Instances.

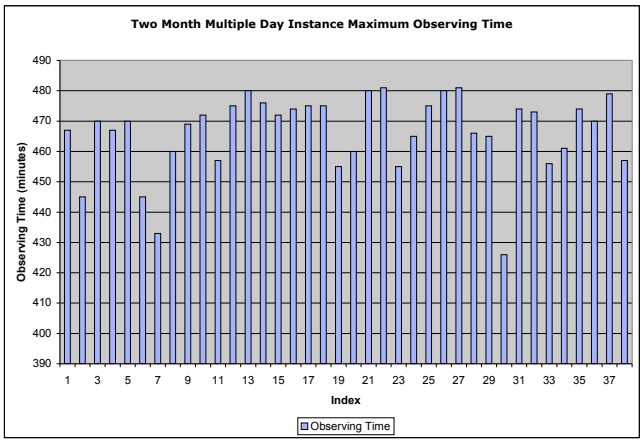

Figure 12: Maximum number of observations scheduled for the Two Month Multiple Day Instances.

matches the Multiple Day Instances we have constructed, we cannot compare the performance of the algorithm on these instances to previously generated plans. We present both the maximum amount of observing time scheduled as well as the maximum flight efficiency, which is the percentage of the flight spent performing observations. This conveys how well the algorithm is able to pack the schedule with observations from the list of candidate objects.

Figure 10 shows the maximum observing time scheduled for each of the one-month Multiple Day Instances. Figure 11 shows the maximum flight efficiencies of the schedules that maximize the observing time. As we see from these figures, the algorithm is able to schedule a considerable amount of observing time for each problem instance, and in most cases the flight efficiencies of these schedules is very high.

Figure 12 shows the maximum observing time scheduled for each of the two-month Multiple Day Instances. Figure 13 shows the maximum flight efficiencies of the schedules that maximize the observing time. Again, we see from these figures, the algorithm is able to schedule a considerable amount of observing time for each problem instance, and in most cases the flight efficiencies of these schedules is very high. We conclude that the algorithm does not become confounded by the presence of many extra observations, and can effectively find good sets of observations to schedule and choose a good ordering for them.

Tuning Algorithm Performance

In conducting our experiments on the Multiple Day Instances, we discovered that computation time became quite large. The main reason for this was the dead leg search. The maximum dead leg duration D, dead leg increment E and heading increment I can all be used to limit the number of prospective dead legs constructed. However, reducing these prospective dead legs may come at a cost in the value of the resulting flight plans. Reducing the maximum dead leg duration to zero may lead to poor plans because some observations may not be possible. Similarly, reducing the dead leg or heading granularity may lead to inefficient plans.

Figures 14 and 15 show the impact of changing D and I on problem instance 19 from the Two Month Multiple Day Instances in Figure 9. For each of the parameter settings

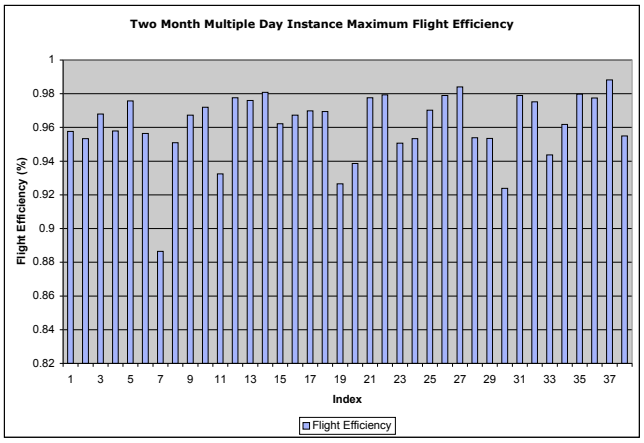

Figure 13: Flight efficiency for the Two Month Multiple Day Instances.

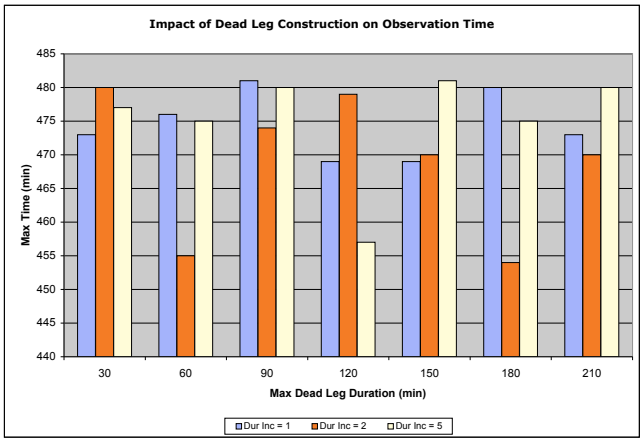

Figure 14: Impact of Dead Leg Construction on Maximum Observing Time Scheduled.

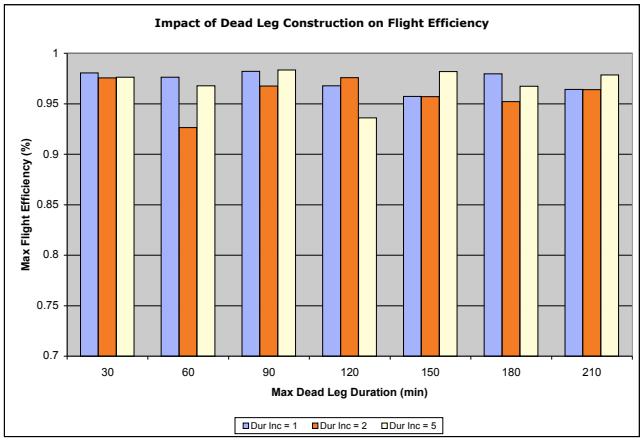

Figure 15: Impact of Dead Leg Construction on Efficiency of Flight Plans Maximizing Number of Observations Scheduled.

we ran 100 samples of the flight planning algorithm. The figure shows the maximum observing time over all flight plans generated. All other algorithm parameters were otherwise the same. We see from this figure that modifying the dead leg search algorithm has some impact on the amount of time spent observing in the best plans found, but that the impact is not very consistent across the range of dead leg construction parameters. The largest difference is 20 minutes, which among the shortest observations over all observation requests in all of the problems. We also see that the impact on the flight efficiency is no more than a few percent at most. We found this to be true across the range of test cases.

We ran an experiment to see how the ForwardPlanner performance scales with increasing numbers of observations. Even though the worst case is $O(KMN^2)$, it is easy to see that performance can vary widely. For instance, the cost of the lookahead phase depends on how many feasible observations there are, and as flight planning progresses this can change dramatically. We took 13 initial states from Moffett Field during August to September of 1995 and used these to create new initial states with between 6 and 102 observations (corresponding to the Two Month Multi-Day instances numbered17 to 28 in Figure 9). We then ran ForwardPlanner 100 times to obtain average performance data. We used a maximum dead leg duration of 60 minutes and a dead leg increment of 2 minutes; otherwise we used the same parameter settings used in the previous experiments. We plot the results in Figure 16. We also plot the results of multiple linear regression to fit a quadratic model of algorithm performance. In this case, we know that M remains roughly constant as N grows. These experiments were run on a 600

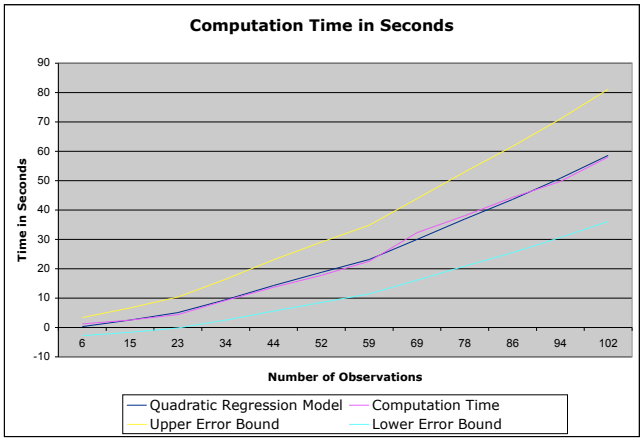

Figure 16: Performance of the algorithm using dead legs of maximum 60 minutes with 2 minute increments.

Mhz Sun Blade with 2 Gigabytes of RAM. We have included the linear regression model, and the models formed by using the upper and lower 95% coefficients This plot shows that the performance of ForwardPlanner is well-modeled by a quadratic function of the number of observations.

Conclusions and Future Work

We have described a telescope observation scheduling problem motivated by the SOFIA General Investigator program. The problem is unique when compared to other scheduling problems in that it involves complex, sequence-dependence constraints governing the simultaneous apparent motion of the objects and the observatory. These are complicated by the requirement that the observatory return to a designated location. We have described a forward search algorithm and associated heuristics, and demonstrated that the resulting search algorithm performs well on a realistic benchmark problem.

The problem described in this paper makes a number of simplifying assumptions. The planner does not account for restricted airspace, which can influence the order of the observations and the characteristics of dead legs. Aircraft motion depends on wind direction and velocity, which are not modeled. In addition, astronomers may impose additional requests on observations, such as a minimum observing altitude, and constraints between calibration objects and actual science objects. Finally, the flight duration approximates the fuel consumption profile of the aircraft. Accounting for these factors will likely require modification of the algorithm described here. In addition, the algorithm described is the first automated solution to the SFPP. There are a large variety of algorithm modifications that are worth exploring, such as modifying the heuristics weights on the fly during search, modifying the lookahead scheme, and implementing local search algorithms to solve this problem. These enhancements are being considered.

Acknowledgments

We would like to thank Sean Casey, the SOFIA Facility Instruments Program Chief Scientist, for the time he has spent describing this problem. We would like to thank Sean Colgan and Allan Meyer, former KAO astronomers and members of the SOFIA staff, for helping mine the KAO archives and providing us with benchmark problems. Lastly, we would like to thank Michael Gross of USRA for his advice and assistance. This work was funded by the NASA Intelligent Systems Program and the NASA Research Technology Objectives and Plans program.

References

Bresina, J. 1996. Heuristic-biased stochastic sampling. In *Proceedings of the 13th National Conference on Artificial Intelligence*.

Ferziger, J. 1981. *Numerical Methods for Engineering Applications*. John Wiley and Sons.

Frank, J., and Jónsson, A. 2003. Constraint-based attribute and interval planning. *Journal of Constraints* To Appear.

Johnston, M., and Miller, G. 1994. Spike: Intelligent scheduling of the hubble space telescope. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling*. Morgan Kaufmann Publishers.

Meeus, J. 1991. *Astronomical Algorithms*. Willmann-Bell, Inc.